Lab 2 Specification – Exploring Performance Assessment
Due (via your git repo) no later than 2 p.m., Friday, 23rd September 2020.
50 points

Lab Goals

- Practice performance assessment problems.
- Reflection on C programming and low-end computing.
- More practice with C programming.

Summary

You will do a few exercises just to refresh your understanding and practice Performance Assessment techniques discussed in class, and extend the learning from in-class activities. We will watch a video clip, which is a segment of an interview with the famous scientist, and one of the inventors of the C programming language, Brian Kernighan. We will take up an additional programming exercise to further solidify our knowledge in C programming.

Learning Assignment

If you have not done so already, please read all of the relevant ”GitHub Guides”, available at https://guides.github.com/, which explains how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as ”Mastering Markdown” and ”Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- PH chapter 01 - section 1.3, 1.5, 1.6 - 1.7
- KR chapter 01 - section 1.1
- Class discussion notes, slides, and in-class coding files.

Assignment Details

Now that we have discussed some fundamental principles behind performance assessment at the hardware-level, and got familiarized with the C programming language, it is now time to implement some challenging requirements of performance assessment. In this process, we will also implement a tool using a C program to evaluate the performance based on user-provided metrics. Additionally, we will watch and reflect on a video to learn some intricacies of the C programming language, to retain the knowledge from the class discussions so far.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) is strictly not allowed. Please note that all work done during laboratory sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as lab assignments, skill tests, projects, etc.

At any duration during and/or after the lab session, students are recommended to team up with the Professor and/or the Technical Leader(s) to clarify if there is any confusion related to the items in the lab sheet and/or class materials.

Section 1: Performance Exercise

This section is worth 20 points. The points breakdown is provided below:

- Task 1 = 10 points
- Task 2 = 10 points

Performance assessment is an important technique that provides the right set of tools to compare machines at the hardware level and guide us in the process of identifying the performance of computers. This procedure is critical in comparing two or more machines and identifying the fastest or slowest machine at the hardware level.

To practice and retain knowledge from class discussions, it is required to complete the following tasks, outlined below:

- **Task 1:** From the 5th edition of the Patterson and Hennessy textbook, please answer all parts to Exercise 1.5 on page 55. It is expected that you present your solution as detailed as possible. You can consider, the level of details to be similar to the examples provided in slides. Prepare your solution file using a markdown file and save the file as `sol-1` in your repository.
- **Task 2:** From the 5th edition of the Patterson and Hennessy textbook, please answer all parts to Exercise 1.7 on page 56. It is expected that you present your solution as detailed as possible. You can consider, the level of details to be similar to the examples provided in slides. Prepare your solution file using a markdown file and save the file as `sol-2` in your repository.

Section 2: Why C?

This section is worth 15 points. The points breakdown is provided below:

- Task 3 = 15 points, a maximum of 3 points awarded for each question.

Brian Kernighan is one of the most famous computer scientists, and the brain behind the existence of the C programming language, today. In this section, we will watch a video and reflect on the points discussed in the clip. This reflection is instrumental to further advance our understanding of C programming, and in general to foster the learning from our recent discussions on Performance Assessment and understand Why C is so influential from the man who invented the language. To complete this part, it is required to do the following:

- **Task 3:** Watch this 10-minute video, by using the link below:

  `https://www.youtube.com/watch?v=ci1PJexnfNE`

  After watching the video, create a markdown file and name it as `video-reflection`. In this file, provide a detailed description of the questions presented below:

  1. Why is C so powerful? and what features are provided in the language to help provide better performance according to the speaker?
  2. What is Moore's law and Why did the speaker mention this popular (law in the field of computer science) in this video?
  3. What is the popular phrase "If ain't broken, then don't fix it" has to do with this video?
  4. What is an assembler and an assembly language? (Research into this online), and what is the speaker's view on high-level languages compared to Assemblers? Note: We will discuss in detail about assembly language in this course.
  5. What did the speaker say about Java? and what programming features were discussed as an example, to compare Java with C? Note: We will discuss in detail about these features in this course.

Section 3: Performance Evaluation Tool

This section is worth 15 points. The points breakdown is provided below:

- Task 4 = 15 points

In this section, we will develop to tool to evaluate the performance at the hardware-level based on user-provided metrics. To complete this part, it is required to have a solid understanding of the concepts discussed in class related to Performance Assessment. It is important to review the slides, reason through and understand the formulas discussed, and also complete the reading assignments as required. There is a C program provided in the starter-code provided repository, to help stay focused on the implementation details, and to get started with the development of this tool. To complete this part, it is required to do the following:

- Review the starter-code in a file named `evaluation.c`. Note: The starter-code has all the necessary prompts to get the user input.
- **Task 4** The start_clock method is in-complete in the starter code. We are required to develop the right set of code in this method, to make the code fully functional. It is required to implement the following requirements:
  1. Set up the code to iterate, through the classes for each sequence. Hint: Nested for-loop.
  2. Find the average CPI for each sequence by invoking the prompt_cpi_perclass and the prompt_inscount_perclass methods.
  3. Find the Total CPU Time taken to execute the instructions for each corresponding sequence. Note: we discussed a formula and an example in class, to find the CPU time based on average CPI. It is available in our lesson-3 part-2 slides.
  4. Lastly, using a simple comparison check, through an if-else logic find the sequence that took the longest cpu time and the sequence that took the shortest cpu time.
  5. Note: represent the CPU time as a float variable and use the **(%.2f notation)** while doing the printf statements to show only two decimal points in the CPU Time.
  6. A segment of the output screen, from executing the program during a sample run, is shown below for your reference. It is not required to match my output line by line, but the expected output should include the CPU time for all the sequences, and an indication of the shortest and longest running sequence. Note: It is perfectly fine to strictly enter integer inputs for all prompts. In other words, the program is expected to handle only integer inputs during the input prompts.

```
        Seq1 took:12.86
        Seq2 took:8.56
        Seq3 took:14.86
        The longest sequence is: Seq3
        The shortest sequence is: Seq2
amohan@ALDENV8075 lab2-code %
```

Submission Details

For this assignment, please submit the following to your GitHub lab repository.

1. **video-reflection** markdown file
2. upload of **sol-1** file
3. upload of **sol-2** file
4. updated version of **evaluation.c** file
5. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is made before the deadline. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine unless otherwise cited.” in all your deliverables such as source code and PDF files.
6. It is recommended to upload a summary file, with the details that you would like the Professor to know while grading the work. For example, it may be reflection of your experience in the lab by highlighting some of the challenges that you had faced and a brief mention of how you addressed those challenges while implementing this lab. The summary file may also include a brief mention of any details that one should know about executing your program and what to expect during the execution.

Grading Rubric

1. Details including the points breakdown are provided in the individual sections above.
2. If a student needs any clarification on their lab credits, it is strongly recommended to talk to the Professor. The lab credits may be changed if deemed appropriate.